CS3025 Compiladores

Semana 5:
Ejercicios
11 Septiembre 2023

Igor Siveroni

Temas para Examen 1

- Expresiones Regulares
- Automatas
- Analisis Lexico / Codigo Scanner
- Gramaticas Libres de Contexto
  - Gramaticas Ambiguas
  - Recursión por la izquierda / Eliminación
  - Factorización por Izquierda
  - Asociatividad (por la izquierda y por la derecha)
  - EBNF: Extended Backus-Naur Form
- Analisis Sintactico / Codigo Parser

Expresiones Regulares y Automatas

1) Elaborar un autómata que acepte los siguientes lenguajes sobre el lenguaje binario:

$a)$ $(10)^*1$

$b)$ 1^*001^*

$c)$ $(11+01)^*1$

$d)$ $1(10)^*1+01^*$

$e)$ $(1+0)^*11$

2) Elaborar un AFD para los siguientes lenguajes sobre el lenguaje binario:

$a)$ $\{w \mid \text{la cadena no contiene a 11 ni a 00}\}$

$b)$ $\{w \mid \text{la cadena no contiene a 10 ni a 01}\}$

$c)$ $\{w \mid \text{la cadena contiene a 100}\}$

Expresiones Regulares y Automatas

3) Elaborar un AFD que acepte los siguientes lenguajes sobre el lenguaje binario:

- $L = \{w \mid w \text{ comienza a con 1 y termina con } 0\}$
- $L = \{w \mid w \text{ no tiene como subcadena a 00 y a 11}\}$

4) Dada la expresión regular (10)* + 1*

- Elabore un AFN que acepte todas las cadenas generadas por la expresión regular.
- Elabore la tabla de transición del AFD relacionado al AFN determinado por la RE anterior.

4.5) El símbolo ^ se usa para denotar complemento de un conjunto. Por ejemplo [^a] denota todos los caracteres del alfabeto menos 'a'. Especificar una RE para definir cadenas entre comillas y otra para definir comentarios de una sola línea.

5) Consideremos cadenas de secuencias de dígitos decimales. Generar expresiones regulares para los siguientes casos

- Numeros con el valor 42
- Numeros que no tengan el valor 42
- Numeros con valor mayor que 42.

Gramaticas y Analisis Sintactico

6) Considere el alfabeto Σ = { a, b }. Describa las gramáticas para obtener los siguientes lenguajes:

$$\{a^i b | i \geq 0\}$$

$$\{a^{2i} b^j | i, j \geq 0\}$$

$$\{(ab)^i b^{j+2} | i, j \geq 0\}$$

$$\{a^i b^i a^{2j} | i, j \geq 0\}$$

7) Demostrar que las siguientes gramáticas son ambiguas:

$a)$

- $S \rightarrow A|B$
- $A \rightarrow aA|b$
- $B \rightarrow Bd|a$

$b)$

- $E \rightarrow E + E|E * E|I$
- $I \rightarrow a|b|c$

Gramaticas y Analisis Sintactico

8) Dada la gramática

$$A \rightarrow AA|(A)|\epsilon$$

- Describa el lenguaje que define.
- Demostrar que la gramática es ambigua

9) La siguiente gramática genera todas las expresiones regulares sobre alfabeto de letras:

$$rexp \rightarrow rexp\ ''|''\ rexp \mid rexprexp \mid rexp\ ''*''\ \mid ''('' rexp\ '')'' \mid letra$$

- Proporcione una derivación para la expresión regular (ab | b)*
- Muestre que la gramática es ambigua

Gramaticas y Analisis Sintactico

10) Considere el alfabeto Σ = { a, b, c }

- Determine la gramática que genere las cadenas palíndromas en Σ .
- Derive las cadenas aabaa y bacacab.
- Escriba los arboles sintácticos correspondientes a las derivaciones de la pregunta anterior.

11) Es ambigua la siguiente gramática? Describir el lenguaje generado por la gramática. Podemos demostrar por inducción que la sub-cadena *ba* no puede ser parte de ninguna cadena del lenguaje?

$$S \rightarrow aS|Sb|a|b$$

12) Demuestre que la gramática definida por la siguiente regla es ambigua:

$$S \rightarrow aS|aSbS|\epsilon$$

Gramaticas y Analisis Sintactico

13) De acuerdo a cada gramática presentada a continuación, indique que tipo de cadenas son aceptadas, mencione si son ambiguas o no, y en caso sean ambiguas, proponga una versión no ambigia

$$a)\ \ A \rightarrow b|AA$$

$$b)\ \ A \rightarrow b|AA|\epsilon$$

14) Escriba una gramatica no ambigua para expresiones booleanas que incluya las constantes *True* y *False*, los operadores *and*, *or* y *not*, además de los paréntesis.

Gramaticas y Analisis Sintactico

15) Considere la gramática

$$\text{lexp} \rightarrow \text{atom} \mid \text{list}$$

$$\text{atom} \rightarrow \text{numero} \mid \text{identificador}$$

$$\text{list} \rightarrow (\text{lexseq})$$

$$\text{lexseq} \rightarrow \text{lexpseq lexp} \mid \text{lexp}$$

- Que lenguaje define?
- Evaluar la cadena (a b (2) (c)) - evaluar es otra manera de decir “buscar una derivación”
- Derive por la izquierda y derecha la cadena (a 23 (m x y))
- Eliminar la recursión por la izquierda
- Evaluar la cadena (a b (2) (c))

Gramaticas y Analisis Sintactico

16) Considere la gramática

$$\text{declaracion} \rightarrow \text{tipo var-list}$$

$$\text{tipo} \rightarrow \text{int} \mid \text{float}$$

$$\text{var-list} \rightarrow \text{identificador, var-list} \mid \text{identificador}$$

- Que lenguaje define?
- Factorizar a la izquierda
- Evaluar la cadena *int x, y, z*

Gramaticas y Analisis Sintactico

Considere la gramática de expresiones aritméticas usada en los laboratorios:

```
Exp ::= Exp (+|-) Term | Exp
Term ::= Term (*|/) Fexp | Fexp
Fexp ::= Factor  '**' Fexp  | Factor
Factor ::= num | '(´ Exp ')'
```

- Realizar las derivaciones a la izquierda y a la derecha que generan 5 + 2**3**4. Los arboles de sintaxis son iguales? La gramática es ambigua?
- Que cambios se necesitan hacer a la gramática para poder ser analizada usando el método de descenso recursivo?
- Escribir un analizador sintáctico para el análisis de expresiones definidas por la nueva (equivalente) gramática.

Gramaticas y Analisis Sintactico

La nueva gramática escrita en EBNF se escribe:

```
Exp ::= Term ((+|-) Term)*
Term ::= Fexp ((*|/) Fexp)*
Fexp ::= Factor  ['**' Fexp]
Factor ::= num | '(' Exp ')'
```

- Notar que la asociatividad por la izquierda de Exp y Term es implementada por el analizador sintáctico al momento de construir el árbol sintáctico o AST.

Extendamos la sintaxis para incluir variables, sentencias para asignar valores a variables e imprimir expresiones listas de sentencias, y listas de sentencias.

```
StmList ::= Stm (; Stm)*
Stm ::= id = Exp | print '(' Exp ')'
Exp ::= Term ((+|-) Term)*
Term ::= Fexp ((*|/) Fexp)*
Fexp ::= Factor  ['**' Fexp]
Factor ::= num | '(' Exp ')' | id
```

Gramaticas y Analisis Sintactico

Tomando en consideración la ultima gramática, extenderla para incluir expresiones condicionales de la forma

```
if-exp ´(´ Cexp , Exp, Exp ´)´
```

Donde, por ejemplo

```
x = 2;
y = if-exp(x > 4, 10, 20);
print(x+y)
```

Imprime 22.

- Escribir el código del analizador léxico

Laboratorio 5

Tomando en consideración la ultima gramática, extenderla para incluir expresiones condicionales de la forma

```
if-exp ´(´ Cexp , Exp, Exp ´)´
```

- La ultima versión del analyzador sintactico e interprete puede encontrarse en (semana 5) los archivos imp.hh, imp.cpp, imp_parser.hh, imp_parser.cpp e imp_test.cpp
- Que pasos son necesarios para implementar `if-exp` ?

- Cual es el problema si queremos extender la gramática para poder permitir sentencias como esta?

  y = x > 4
- Como podemos solucionarlo?